

REMARKS

Claims 1-14 stand rejected under 35 U.S.C. § 102(e) as being allegedly anticipated by U.S. Patent Application Publication No. 2003/0046572 to *Newman et al.* (hereinafter "*Newman*").

In this Response, the specification is amended to correct informalities. No claims are amended. No claim is canceled. Accordingly, claims 1-14 are pending in the present application. Applicants respectfully request reconsideration of the application in view of the above amendments and remarks made herein.

I. Rejections Under 35 U.S.C. § 102

Claims 1-14 stand rejected under 35 U.S.C. § 102(e) as being allegedly anticipated by *Newman*, for the reasons set forth on pages 2-4 of the Office Action. Applicants respectfully traverse this rejection.

With respect to claims 1 and 8, Applicants respectfully submit that *Newman* does not anticipate these claims because *Newman* does not disclose "providing a security dictionary comprising one or more security catalogs; ... generating a working encryption key; internally encrypting said working encryption key using a public key from an authorized user; storing said encrypted working key in a security catalog; and using said working key to internally encrypt said data," as in claims 1 and 8.

Applicants' specification discloses "[t]he approach of the invention is to add *encryption as an internal database feature* so that data can be stored in encrypted form and can also be processed efficiently and securely. ... Because efficient *database techniques such as indexes and query optimization are integrated together with the*

encryption function, the efficiency of the RDBMS remains available to the user,” paragraph 26, Pub. No. 2003/0123671 (emphasis added). One feature of the present invention is a security dictionary; before any database object is altered, the security dictionary will be checked first. The security dictionary of the claimed invention comprises one or more security catalogs. A security catalog can never be updated manually by anyone, and its access is controlled by a strict authentication and authorization policy. Another feature of the present invention is a working encryption key. The working encryption key, which is *internally encrypted* and stored in a security catalog, is used to *internally encrypt* data.

In regard to *Newman*, Applicants note that it discloses “the invention comprises two major components. First, the invention comprises a low-level API, which functions as a shell, providing cryptographic algorithms for Procedural Language/Structured Query Language (PL/SQL) developers. Second, the invention comprises a key management system which utilizes the low-level API to provide a turnkey solution to automatically and transparently encrypt data in columns and rows.” (Paragraph 21.) Therefore, *Newman* provides *encryption service to end users via add-on APIs*. In contrast, the present invention provides *encryption as an internal database server core feature*.

Applicants note that DbEncrypt™, referred to in *Newman* (paragraphs 28 and 33) as “the DBENCRYPT package” and relied upon by the Examiner for the assertion that *Newman* teaches “providing a security dictionary,” comprises a variety of encryption algorithms, templates and a GUI. (DbEncrypt™ supports Oracle and Microsoft SQL Server.) Applicants respectfully submit that “the DbEncrypt package,” does not teach “a

security dictionary comprising one or more security catalogs,” as in claims 1 and 8.

Therefore, for at least this reason, *Newman* does not anticipate claims 1 and 8.

Moreover, Applicants submit that “the DBENCRYPT_KEYS table,” in *Newman* (paragraphs 32, 34 and 35), does not teach “one or more security catalogs,” as in claims 1 and 8. Additionally, Applicants submit that, because *Newman* provides *encryption service to end users via add-on APIs*, *Newman* can not teach “*internally encrypting said working encryption key ... and using said working key to internally encrypt said data*,” as in claims 1 and 8 (emphasis added).

For at least the above reasons, *Newman* does not anticipate claims 1 and 8. Applicants respectfully submit that inasmuch as claims 2-7 are dependent on claim 1 and claims 9-14 are dependent on claim 8, and claims 1 and 8 are patentable over *Newman*, claims 2-7 and 9-14 are patentable as dependent on patentable independent claims. Withdrawal of the instant rejections is respectfully requested.

With regard to claims 4 and 11, Applicants submit that nowhere does *Newman* teach “wherein said step of associating said data with a database column and a user is accomplished with an *extended SQL syntax*,” as in claims 4 and 11 (emphasis added). Applicants note that since a change in SQL syntax of itself requires change in the database server kernel, *Newman* clearly does not teach this element because *Newman* only provides *encryption service to end users via add-on APIs*.

With respect to claims 5 and 12, Applicants submit that nowhere does *Newman* teach “wherein said working key is provided by a user,” as in claims 5 and 12. To the contrary, *Newman* (paragraph 37) discloses “when the user attempts to access encrypted data, the encrypted data key (1) for the current user is retrieved from the

DBENCRYPT_KEYS table (2), and is decrypted ... (3) using the private key (4) stored in the application context," which altogether precludes a reasonable assertion that *Newman* teaches "said working key is provided by a user," as in claims 5 and 12.

In the Office Action, with regard to claim 7 and 14, the Examiner asserts that *Newman* "discloses the claimed limitation wherein the steps of: receiving a query and private key from a user; checking the ownership of an encrypted column using said security catalog to verify the user is authorized; internally decrypting said encrypted working encryption key with said private key; internally decrypting said encrypted column with said working key; processing said query; and returning an answer to said query to the user (See page 4, Sections 0072-0080, page 5 Sections 0081-0089)." Office Action mailed July 1, 2005, on page 4. Applicants disagree and respectfully submit that nowhere in *Newman* does it teach "receiving a query and private key from a user; ... internally decrypting said encrypted working encryption key with said private key; internally decrypting said encrypted column with said working key; processing said query; and returning an answer to said query to the user," as in claims 7 and 14. The cited passages, specifically, page 4, sections 72-80, page 5, sections 81-89, as relied upon by the Examiner, disclose:

[0072] When a column is selected to be encrypted, the following steps are taken. [0073] A symmetrical key is created. [0074] The column is encrypted with the symmetrical key (5). [0075] For each user that should have access to the table, a copy of the key is encrypted with the public key and stored in the DBENCRPYT_KEYS table (2). [0076] The table is renamed with the _BASE extension. [0077] A view is created to decrypt data read from the table. [0078] A trigger is created on the view to encrypt data written to the table. [0079] Application context are built-in features of Oracle. There are unique problems that

the invention solves in using application contexts to store Encryption keys. Although application contexts are limited to 255 characters, the invention solves this problem by assembling the key into segments that can be stored in separate contexts and then reassembling the key when necessary. [0080] Values in an application context are referenced using user defined parameter names. When storing a value in the context, the user specifies the parameter name and the value of the parameter. When retrieving a value, the parameter name is all that is required. If the parameter name is valid, then the appropriate value is returned to the user; otherwise a NULL value is returned.

Disassembling (or Partitioning) keys in the context: The algorithm for partitioning the keys into 256 is as follows: [0081] Determine how many 256-character segments there are by taking ceiling of the length of the key divided by 256. The ceiling of a number is defined as the first integer that is greater or equal to the given value. [0082] Store the number of segments obtained in the previous step in the user context for later use when disassembling (or reconstituting) the key. [0083] Initialize a pointer to the first character of the key. [0084] Loop through the number of segments determined in the first step. In each loop, retrieve 256 characters of the key starting at the pointer and store them in the invention's context, then update the pointer's position by 256 characters. [0085] Retrieving (or reconstituting) the key consists of somewhat the reverse of the partitioning process: [0086] Retrieve the number of segments that we stored in the invention's context. [0087] Check that the value is valid, i.e. not null or greater than 0. [0088] Initialize a variable that will hold the reconstituted key. [0089] If valid then loop through the number of segments. In each loop we retrieve from the context each segment of the key and append it to the variable.

However, Applicants found no teaching of the instant claim language in the cited passages as relied upon by the Examiner in rejecting claims 7 and 14.

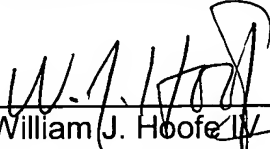
Withdrawal of the rejections under 35 U.S.C. § 102(e) is respectfully requested.

CONCLUSION

In view of the foregoing, it is believed that all claims now pending patentably define the subject invention over the prior art of record and are in condition for allowance. Issuance of a Notice of Allowance is respectfully requested.

Respectfully submitted,

Dated: October 3, 2005



William J. Hoofe
Reg. No. 54,183
Attorney for Applicants

F. CHAU & ASSOCIATES, LLC
130 Woodbury Road
Woodbury, New York 11797
Tel: (516)-692-8888
Fax: (516)-692-8889